BAB 2

LANDASAN TEORI

2.1. Teori Umum

2.1.1. Pengertian Sistem

Sistem adalah sebuah kumpulan dari komponen-komponen yang saling berhubungan, dengan sebuah batasan yang jelas, saling bekerjasama satu sama lain untuk mencapai sebuah tujuan tertentu (O'Brien & Marakas, 2011, p. 4).

2.1.2. Pengertian Data

Data adalah komponen yang paling penting dalam *Database Management System*, berasal dari sudut pandang pengguna akhir. Data bertindak sebagai jembatan yang menghubungkan antara mesin dengan user (Connolly & Begg, 2005, p. 20).

2.1.3. Pengertian Informasi

Informasi merupakan data yang sudah diolah menjadi suatu konteks yang memiliki manfaat dan arti untuk dimanfatkan oleh pengguna akhir (O'Brien & Marakas, 2011, p. 34).

2.1.4. Pengertian Perangkat Lunak

Perangkat lunak (*software*) adalah instruksi-instruksi dari program komputer yang ketika dieksekusi akan menyediakan fitur, fungsi, dan performa yang diinginkan (Pressman, 2010, p. 4).

2.1.5. Pengertian Sistem Informasi

Sistem informasi merupakan kombinasi dari manusia, perangkat keras, perangkat lunak, jaringan komunikasi, sumber data, dan kebijakan serta prosedur dalam hal menyimpan, mendapatkan, mentrasformasikan, dan menyebarluaskan informasi di dalam sebuah organisasi (O'Brien & Marakas, 2011, p. 4).

2.1.6. Teori Basis Data

2.1.6.1. Pengertian Basis Data

Basis data (*database*) merupakan sebuah koleksi dari data yang berelasi secara logika, dan sebuah deskripsi dari data tersebut, dedesain untuk memenuhi kebutuhan akan informasi oleh sebuah organisasi (Connolly & Begg, 2005, p. 15).

Dalam konsep basis data terdapat *table* yang berisikan *field* dan *record*. *Field* merupakan implementasi fisik dari sebuah atribut data. *Field* adalah unit terkecil yang memiliki arti dari data yang disimpan dalam basis data (Whitten & Bentley, 2007, p. 521)

Terdapat empat macam *field* yang dapat disimpan dalam basis data (Whitten & Bentley, 2007, p. 521).

a. Primary Key

Primary key adalah *field* yang memberikan tanda unik dalam setiap *record*.

b.Secondary Key

Sebuah *field* yang mengidentifikasikan sebuah *record* tunggal atau bagian dari beberapa *record* disebut *secondary key*.

c. Foreign Key

Foreign key adalah *field* yang menunjuk ke sekumpulan *record* yang terdapat dalam *table* yang lain di basis data.

d. Descriptive Field

Descriptive Field adalah *field* selain *key fields* yang menyimpan data bisnis.

Records adalah sebuah koleksi *fields* yang diatur dalam format yang telah ditentukan sebelumnya (Whitten & Bentley, 2007, p. 521).

2.1.6.2. *Database Management System*

Database Management System (DBMS) merupakan perangkat lunak yang berinteraksi dengan pengguna dari program dan basis data itu sendiri. Pada umumnya, sebuah DBMS menyediakan fasilitas berikut ini (Connolly & Begg, 2005, pp. 16-17):

- Memungkinkan pengguna untuk mendefinisikan basis data, biasanya menggunakan *Data Definition Language* (DDL). DDL memungkinkan pengguna untuk menentukan tipe data, struktur, dan *constraint* pada data untuk kemudia disimpan ke dalam *database*.
- Memungkinkan pengguna untuk memasukkan, memodifikasi, menghapus, dan mendapatkan data dari tempat penyimpanan, biasanya dengan menggunakan *Data Manipulation Language* (DML). Memiliki tempat penyimpanan pusat untuk seluruh data dan deskripsi data memungkinkan DML untuk menyediakan sebuah fasilitan pengolahan data, dikenal dengan bahasa *query*. Penggunaan bahasa *query* menyelesaikan berbagai masalah yang biasa ditemui pada sistem berbasis *file* di mana *user* harus bekerja dengan perangkat lunak khusus sesuai dengan jenis *file* yang terkait. Bahasa *query* yang paling umum saat ini adalah *Structured Query Language* (SQL), yang mana menjadi bahasa standar untuk DBMS relasional.
- Menyediakan akses kontrol ke dalam basis data. Sebagai contoh, dapat menyediakan:
    - sistem keamanan, yang mencegah pengguna yang tidak memiliki izin untuk mengakses basis data;

- sebuah sistem integritas, yang mengelola konsistensi data yang tersimpan;
- sebuah sistem kontrol konkurensi, yang memungkinkan akses bersamaan ke dalam basis data;
- sebuah sistem kontrol *recovery*, yang mengembalikan basis data ke kondisi sebelum terjadinya kegagalan perangkat lunak;
- sebuah katalog, yang berisi deskripsi dari data yang tersimpan di dalam basis data

2.1.6.3. *Structured Query Language*

- **Pengertian SQL**

  Structured Query Language (SQL) merupakan bahasa *query* standar yang digunakan untuk mengakses basis-data relasional. (Krieger, 2011)

- **Kelompok *Statement* dalam SQL**

  - ***Data Definition Language* (DDL)**

    DDL adalah bahasa yang memperbolehkan seorang *database adminitrator (DBA)* atau *user* untuk mendeskripsikan nama dari suatu entitas, atribut, hubungan yang dibutuhkan oleh aplikasi bersamaan dengan integritas data dan keamanan datanya (Connolly dan Begg, 2005, p40).

DDL merupakan kelompok perintah yang berfungsi untuk mendefinisikan atribut-atribut basis-data, tabel, kolom, batasan-batasan terhadap suatu attribut serta hubungan antar tabel.

Statement DDL antara lain *CREATE TABLE, ALTER TABLE, DROP TABLE, CREATE INDEX*, dan *DROP INDEX*

.

- ***Data Manipulation Language* (DML)**

  DML merupakan bahasa yang memberikan fasilitas pengoperasian data yang ada dalam basis-data (Connolly dan Begg, 2005, p41). Pengoperasian manipulasi data diantaranya memasukkan data baru, mengubah data, maupun menghapus data dari basis-data.

  Terdapat dua tipe DML, yaitu:

  - *Procedural DML*

    DML prosedural merupakan bahasa yang mengizinkan pengguna (*programmer*) dalam memberikan instruksi kepada sistem mengenai data apa yang dibutuhkan dan bagaimana cara pengambilan datanya. Dengan kata lain, pengguna harus menjelaskan operasi pengaksesan data dengan menggunakan prosedur yang ada untuk mendapatkan informasi yang dibutuhkan.

  - *Non-Procedural DML*

    DML *non-prosedural* merupakan bahasa yang memungkinkan pengguna untuk menentukan data apa yang dibutuhkan dengan

menyatakan spesifikasinya tanpa menjelaskan bagaimana prosedur yang harus dilaksanakan untuk mendapatkannya.

- ***Data Control Language*** **(DCL)**

  Dalam DCL berisi perintah-perintah untuk mengendalikan pengaksesan data. Pengendalian dapat dilakukan pada setiap pengguna, tabel, kolom, ataupun operasi yang boleh dilakukan. Perintah-perintah ini berkaitan dengan manajemen keamanan basis-data. Perintah-perintah yang termasuk diantaranya *GRANT, REVOKE,* dan *LOCK TABLE.*

- **Pengendali Transaksi**

  Pengendali transaksi merupakan perintah-perintah yang berfungsi untuk mengendalikan pengeksekusian suatu transaksi. Perintah yang termasuk dalam DCL diantaranya *COMMIT,* dan *ROLLBACK*

- **Pengendali Programatik**

  Dalam kelompok statemen ini, perintah-perintah yang termasuk dalam pengendali programatik merupakan pernyataan-pernyataan yang berhubungan dengan pemanfaatan SQL dalam bahasa lain. Pernyataan-pernyataan terdapat pada bahasa konvensional generasi ketiga (3-GL), seperti COBOL. Perintah-perintah yang termasuk dalam kategori ini diantaranya *CLOSE, DECLARE, FETCH,* dan *OPEN.*

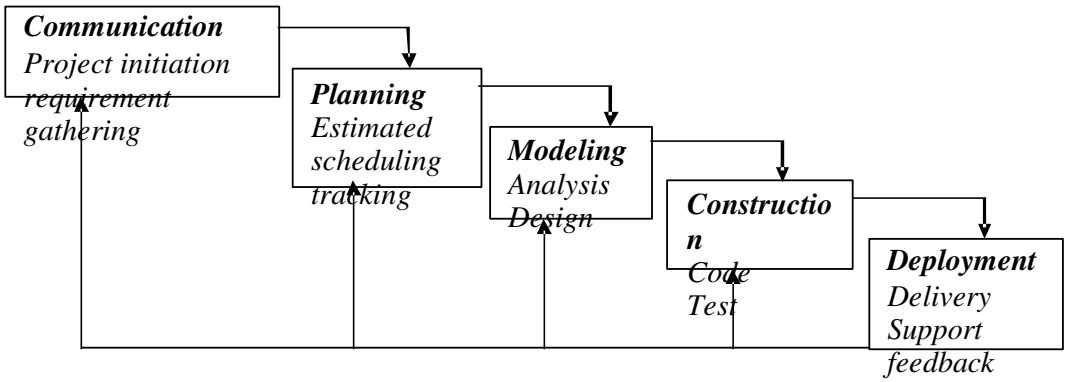
2.1.7. Teori *Development Software*

2.1.7.1. Metode Perancangan SDLC

Systems Development Life Cycle (SDLC) dalam rekayasa sistem dan rekayasa perangkat lunak, adalah proses pembuatan dan pengubahan sistem serta model dan metodologi yang digunakan untuk mengembangkan sistem-sistem tersebut. Metode Perancangan SDLC yang digunakan dalam penulisan ini adalah Model Proses *Waterfall*. (Pressman, 2010, p. 39).

- Model Proses *Waterfall*

  Model *waterfall* adalah proses pengembangan *software* sekuensial, dimana kemajuan dipandang sebagai terus mengalir ke bawah (seperti air terjun) melalui tahapan konsepsi, inisiasi, analisis, desain, konstruksi, pengujian dan pemeliharaan



Gambar 2.1 *Waterfall Model*

Berikut adalah penjelasan dari tahap-tahap yang dilakukan di dalam model *waterfall* :

- ***Communication*.** Permodelan ini diawali dengan komunikasi dan kolaborasi dengan konsumen (*stackholders*) untuk mencari kebutuhan dari keseluruhan sistem yang akan diaplikasikan ke dalam bentuk *software*. Hal ini sangat penting, mengingat *software* harus dapat berinteraksi dengan elemen-elemen yang lain seperti *hardware*, *database*, dsb. Tahap ini sering disebut dengan *Project Definition*.
- ***Planning*.** Proses ini menetapkan rencana untuk pengerjaan *software* yang meliputi : tugas-tugas teknis yang akan di lakukan, resiko yang mungkin tejadi, sumber-sumber yang di butuhkan, hasil yang akan di buat, dan jadwal pengerjaan.
- ***Modeling***. Proses ini meliputi pembuatan model yang memungkinkan pengembang dan konsumen untuk lebih memahami kebutuhan perangkat lunak dan desain yang akan mencapai kebutuhan tersebut.
- ***Construction***. Proses ini merupakan proses gabungan dari *coding* dan *testing*. Untuk dapat dimengerti oleh mesin computer, maka desain tadi harus diubah bentuknya menjadi bentuk yang dapat dimengerti oleh mesin, yaitu ke dalam bahasa pemrograman melalui proses *coding*. Tahap ini merupakan implementasi dari tahap design yang secara teknis nantinya dikerjakan oleh *programmer*. Sedangkan *testing* adalah sesuatu yang dibuat haruslah diujicobakan. Demikian

juga dengan *software*. Semua fungsi-fungsi *software* harus diujicobakan, agar *software* bebas dari error, dan hasilnya harus benar-benar sesuai dengan kebutuhan yang sudah didefinisikan sebelumnya.

- ***Deployment***. Pemeliharaan suatu *software* diperlukan, termasuk di dalamnya adalah pengembangan, karena *software* yang dibuat tidak selamanya hanya seperti itu. Ketika dijalankan mungkin saja masih ada error kecil yang tidak ditemukan sebelumnya, atau ada penambahan fitur-fitur yang belum ada pada *software* tersebut. Pengembangan diperlukan ketika adanya perubahan dari eksternal perusahaan seperti ketika ada pergantian sistem operasi, atau perangkat lainnya.

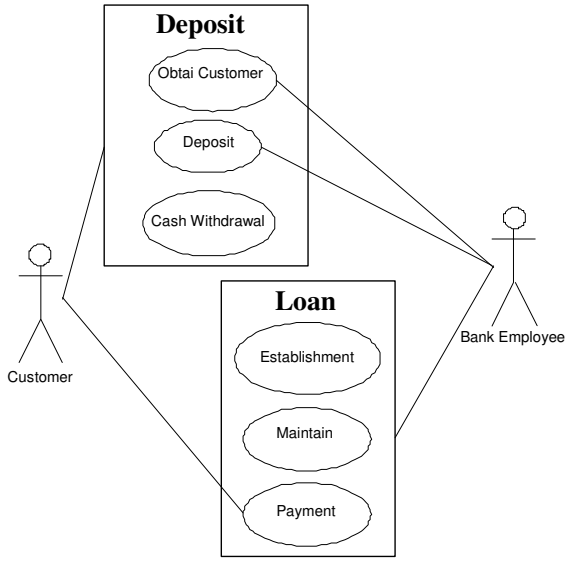
2.1.7.2. *Unified Modeling Language*

Unified Modeling Language (UML) adalah suatu konvensi pemodelan yang digunakan untuk menspesifikasikan atau mendeskripsikan sebuah sistem piranti lunak yang terkait dengan objek. UML terdiri dari beberapa diagram, yaitu *Use-case Diagram*, *Class Diagram*, *Activity Diagram*, dan *Sequence Diagram*. (Whitten & Bentley, 2007, p. 371).

2.1.7.3. *Use-case Diagram*

Use-case adalah pola interaksi antara sistem dengan aktor dalam sebuah domain aplikasi. *Actors* dan *use-case* adalah dua elemen utama

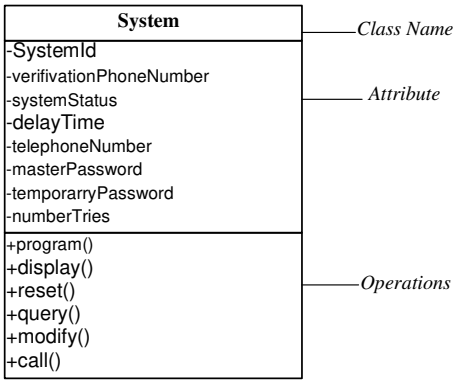
dalam deskripsi. Aktor dan *use-case* dapat terhubung satu sama lainnya, dan dapat saling melengkapi dalam struktur *use-case diagram* (Pressman, 2010, p. 133). Contoh *use-case Diagram:*



Gambar 2.2 *Use Case* Diagram

2.1.7.4. *Class Diagram*

Class Diagram menjelaskan kumpulan dari *class* dan hubungan diantaranya yang struktural. Diagram ini menunjukkan kelas objek yang menyusun sistem dan juga hubungan antara kelas objek tersebut (Pressman, 2010, p. 167).

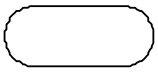


Gambar 2.3 *Class Diagram*

2.1.7.5. *Activity Diagram*

Activity diagram adalah diagram untuk menggambarkan aksi-aksi dan keputusan-keputusan yang terjadi sesuai fungsi-fungsi yang telah dilakukan (Pressman, Software Engineering: A Practitioner's Approach, 2010, p. 161). Komponen-komponen *activity diagram* :

1. *Rounded rectangles*, menggambarkan sistem yang specifik

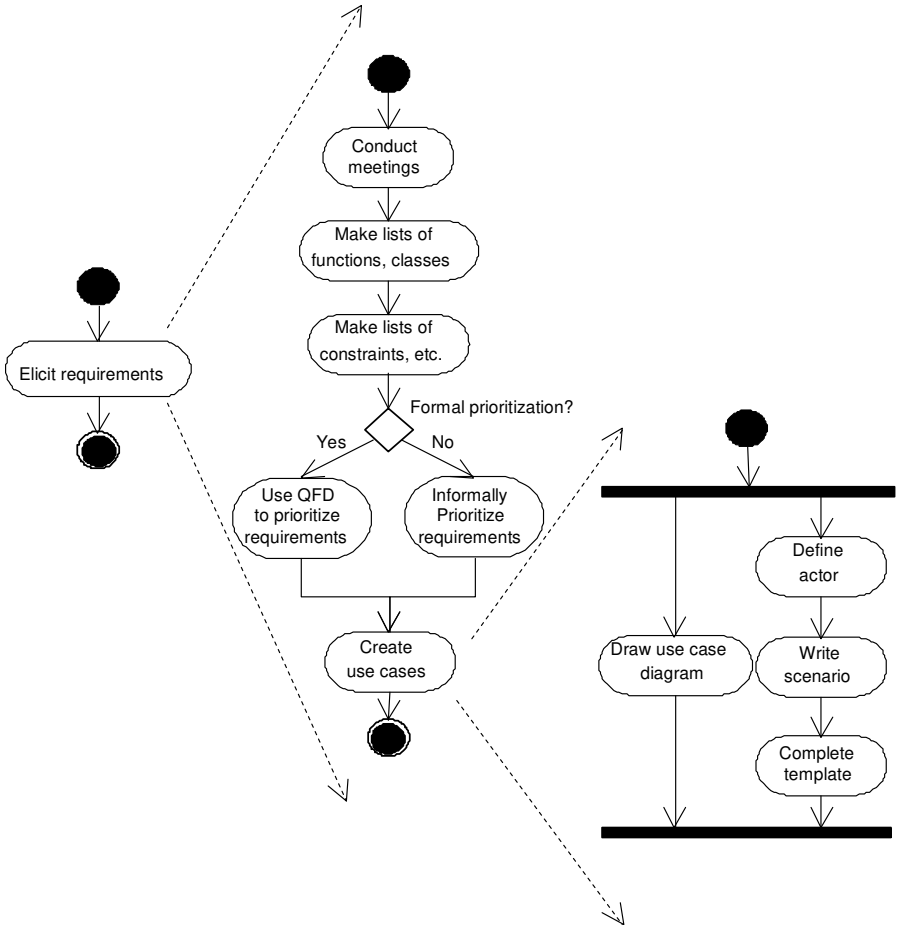


2. *Arrows*, menggambarkan aliran aktivitas dalam sistem

3. *Decision diamond*, menggambarkan suatu pilihan keputusan/ tindakan

4. *Solid horizontal lines*, untuk menggambarkan aktivitas paralel yang terjadi

Contoh *Activity Diagram* :



Gambar 2.4 ***Activity Diagram***

2.1.7.6. *Sequence Diagram*

Sequence diagram menggambarkan bagaimana objek-objek berinteraksi satu sama lain melalui pesan dalam eksekusi dari sebuah *use case* atau operasi. Sebuah system *sequence diagram* membantu untuk mengidentifikasi pesan tingkat tinggi yang masuk dan keluar dari sistem.

Beberapa notasi yang terdapat di dalam *system sequence diagram*, yaitu (Whitten & Bentley, 2007, pp. 394-395):

1. *Actor*, berupa aktor yang memulai pada *use case* ditunjukkan dengan simbol *use case* aktor.

2. *System*, berupa kotak yang menunjukkan sistem yang sedang berjalan, ditandai dengan notasi titik dua (:) pada awal nama sistem.

3. *Lifelines*, berupa garis putus-putus menurun dari simbol aktor dan sistem.

4. *Activation bars*, berupa balok yang terletak di atas *lifelines*, menunjukkan periode waktu ketika terdapat interaksi secara aktif.

5. *Input messages*, berupa anak panah mendatar dari aktor ke sistem yang menunjukkan pesan masuk, dimana kata pertama diawali dengan huruf kecil, sedangkan kata berikutnya diawali dengan huruf besar dan tidak ada spasi, diikuti dengan tanda kurung yang berisi parameter yang dibutuhkan.

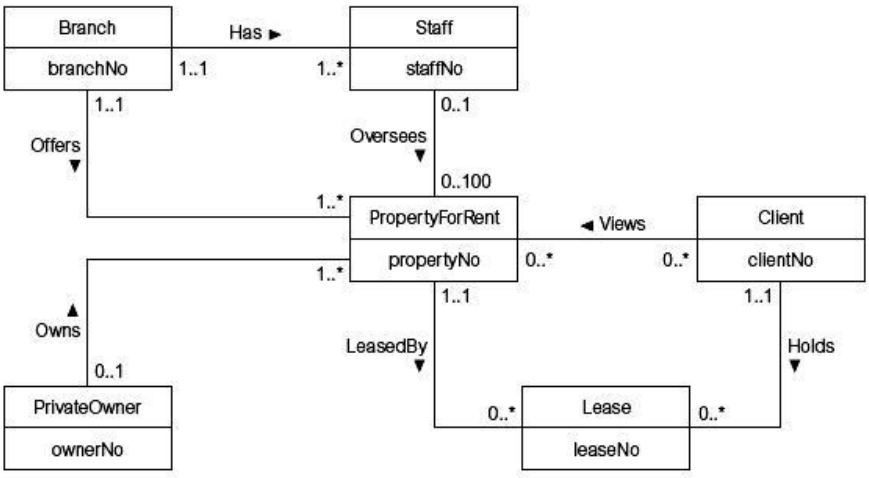
6. *Output messages*, berupa anak panah mendatar dengan garis putus-putus dari sistem ke aktor yang menunjukkan pesan keluar.

7. *Receiver Actor*, berupa aktor lain yang menerima pesan dari sistem.

8. *Frame*, berupa kotak yang menambahkan pesan terpisah untuk menunjukkan perulangan, *alternative*, atau pilihan.

2.1.7.7. *Entity Relationship Diagram* (ERD)

Entity Relationship Diagram (ERD) menggambarkan komponen-komponen dari entitas dan relasi yang masing-masing dilengkapi dengan atribut-atribut yang merepresentasikan seluruh fakta dari keadaan sesungguhnya yang akan kita tinjau. ERD pertama kali dikemukakan oleh Peter Chen untuk mendesain sistem basis data relasional dan kemudian konsep tersebut diperluas oleh penerusnya. Komponen-komponen yang diidentifikasikan oleh ERD adalah objek data, atribut, relaso, dan berbagai tipe indikator. Tujuan utama dari ERD adalah untuk merepresentasikan objek data dan relasinya satu sama lain (Pressman, 2010, p. 166).



Gambar 2.5. Contoh dari *Entity Relationship Diagram*

2.1.8. Teori *Internet*

2.1.8.1. Pengertian *Internet*

Internet adalah koleksi jaringan komputer yang ada di seluruh dunia yang menghubungkan jutaan komputer yang digunakan untuk bisnis, pemerintahan, lembaga pendidikan, organisasi, dan individu menggunakan modem, kabel telepon, kabel televisi, satelit, dan peralatan komunikasi lainnya (Shelly, Woods, & Dorin, 2008, p. 2).

2.1.8.2. Pengertian World Wide Web

World Wide Web adalah bagian dari internet yang mendukung multimedia dan terdiri dari kumpulan dokumen yang terhubung, biasanya lebih umum dipanggil sebagai *web* (Shelly, Woods, & Dorin, 2008, p. 3).

World Wide Web terdiri dari berbagai situs *web* yang memiliki fungsi berbeda tergantung pada pembuat situs *web* tersebut.

Halaman *Web* adalah dokumen-dokumen yang saling terhubung atau halaman berisi informasi yang ada di *Web. Web* selalu berubah-ubah dan terdiri dari ratusan juta halaman *web*, hal ini disebabkan oleh kemudahan dalam pembuatan halaman *web*, dan masih dapat terus bertambah setiap saatnya.

Halaman - halaman*web* yang ada disimpan di *Web Server* yaitu sebuah komputer yang menyimpan dan mengirim halaman-

halaman yang diminta. Setiap komputer yang memiliki perangkat lunak *web server* yang sudah terinstall di dalamnya dan juga terhubung dengan internet, dapat menjadi *web server*, setiap situs yang ada, dijalankan dari satu atau lebih *web server*. Sebuah situs *web* yang cukup besar bisa saja tersebar pada beberapa *server* di lokasi geografis yang berbeda.

Secara umum, ada tiga jenis situs web yaitu internet, intranet, dan ekstranet (Shelly, Woods, & Dorin, 2008, p. 6).

1. Internet

   Internet dikenal juga sebagai situs *web*, adalah sebuah situs yang tersedia untuk umum. Perorangan, grup, perusahaan, dan lembaga pendidikan menggunakan situs internet atau situs *web* untuk bermacam-macam kebutuhan, seperti menjual produk dan jasa, memberikan dukungan teknis dan produk untuk pelanggan, memberikan informasi mengenai suatu perusahaan berdasarkan kebutuhan, dan lain sebagainya.

2. Intranet

   Intranet adalah jaringan khusus yang menggunakan teknologi internet untuk berbagi informasi ataupun data perusahaan diantara para karyawannya. Intranet terletak di dalam suatu jaringan organisasi atau perusahaan. Beberapa intranet juga menggunakan sistem sandi untuk menghindari bahaya dari akses yang tidak berhak.

3. Ekstranet

Ekstranet adalah jaringan pribadi yang menggunakan teknologi internet untuk berbagi informasi ataupun data bisnis dengan mitra perusahaan tertentu. Kebanyakan ekstranet dilindungi oleh sandi untuk membatasi akses kepada *supplier*, vendor, mitra, ataupun pelanggan.

Untuk melihat halaman *web* dalam sebuah situs, pada komputer diperlukan *web browser*, atau umumnya dikenal dengan nama *browser*. *Web browser* adalah sebuah aplikasi yang menerjemahkan, menampilkan halaman web dan memungkinkan pengguna untuk berinteraksi dengan halaman *web*. Contoh *web browser* adalah *Microsoft Internet Explorer, Google Chrome, Opera, Mozilla Firefox*, dan *Apple Safari*.

Browser menyediakan berbagai macam fitur, yaitu kemampuan untuk menemukan halaman *web*, untuk maju dan mundur antara halaman *web*, untuk menandai halaman web yang dikunjungi, dan untuk pengaturan keamanannya.

2.1.8.3. Pengertian *Hypertext Transfer Protocol* dan *Hypertext Transfer Protocol Secure*

Hypertext Transfer Protocol (HTTP) adalah suatu protokol yang digunakan untuk komunikasi atau mengirim informasi oleh WWW. HTTP mendefinisikan bagaimana suatu pesan dapat diformat dan dikirimkan dari *server* ke *client*. *Client* membuat suatu HTTP *request*sedangkan *server* berguna untuk menyimpan dan

membuat *resources* (Pressman & Lowe, Web Engineering a Practitioner's Approach, 2009, p. 268).

HyperText Transport Protocol Secure (HTTPS) memiliki pengertian sama dengan HTTP tetapi dengan alasan keamanan (*security*), HTTPS memberi tambahan *Secure Socket Layer* (SSL). Teknologi HTTPS protokol mencegah kemungkinan "dicurinya" informasi penting (kartu kredit adalah contoh yang paling sering disebut-sebut) yang dikirimkan selama proses komunikasi berlangsung antara *user* dengan *web server* (atau sebaliknya). Secara teknis, *website* yang menggunakan HTTPS akan melakukan enkripsi terhadap informasi (data) menggunakan teknik enkripsi SSL. Dengan cara ini meskipun seseorang berhasil "mencuri" data tersebut selama dalam perjalanan user *web server*, orang tersebut tidak akan bisa membacanya karena sudah diubah oleh teknik enkripsi SSL.

2.1.9. Teori Interaksi Manusia dan Komputer

2.1.9.1. Pengertian Interaksi Manusia dan Komputer

Interaksi Manusia dan Komputer (IMK) adalah disiplin ilmu yang berhubungan dengan perancangan, evaluasi, dan implementasi sistem komputer interaktif untuk digunakan oleh manusia. Titik berat IMK adalah perancangan dan evaluasi antarmuka pemakai (user interface). Antarmuka pemakai adalah bagian sistem komputer yang

memungkinkan manusia berinteraksi dengan komputer (Schneiderman & Plaisant, 2005, p. 4).

2.1.9.2. Delapan Aturan Emas Perancangan *User Interface*

Merancang *user interface* di perlukan suatu aturan tertentu, biasanya dikenal dengan delapan aturan emas (Schneiderman & Plaisant, 2005, p. 74), yaitu:

1. Berusaha untuk konsisten.

   Rangkaian tindakan yang konsisten digunakan dalam keadaan yang terminologinya seperti pada *promp*, *menu* dan layar *help*, warna, tampilan, dan kapitalisasi. Pengecualian dalam pembuatan *password*, tidak boleh berulang.

2. Menyediakan *usability universal.*

   Dalam tingkat penggunaan yang tinggi, *user* cenderung untuk mempercepat interaksi yang dilakukan sehingga perintah khusus menjadi sangat berguna bagi *user*.

3. Memberikan umpan balik yang informatif.

   Tindakan yang dilakukan harus memberikan umpan balik agar *user* mengerti apa yang telah dilakukannya.

4. Merancang *dialog* yang memberikan penutupan (keadaan akhir).

   Keadaan akhir menandakan selesainya suatu kegiatan yang diberitahukan kepada *user* melalui umpan balik. Tanpa adanya

dialog untuk mencapai keadaan akhir maka *user* akan dibingungkan dengan apa yang ia kerjakan.

5. Memberikan pencegahan kesalahan dan penanganan kesalahan yang sederhana.

   Dengan memberikan penanganan kesalahan maka sistem dapat membantu memberikan solusi jika terjadi kesalahan.

6. Memungkinkan pembalikan aksi yang mudah.

   Undo memberikan keleluasaan *user* untuk bergerak dimana jika *user* melakukan kesalahan, *user* dapat kembali tanpa harus cemas rusaknya hal yang sedang dikerjakan.

7. Mendukung pusat kendali internal.

   Dengan pengaturan yang menyeluruh, *user* dapat menggerakkan sistem sesuai kebutuhan mereka dan menggunakan sistem dengan lebih maksimal.

8. Mengurangi beban ingatan jangka pendek.

   Keterbatasan kemampuan manusia memproses informasi dalam jangka waktu yang pendek harus diperhatikan dalam membuat prosedur sehingga tidak menghalangi penggunaan sistem.

2.2. Teori Khusus

2.2.1. *Learning Management System* (LMS)

Learning Management System (LMS) merupakan sebuah aplikasi perangkat lunak yang mengotomatisasi proses administrasi, pengawasan, dan pelaporan dari suatu kegiatan pelatihan atau pembelajaran. Sebuah LMS yang baik harus dapat melakukan hal-hal seperti memusatkan dan mengotomatisasi

proses administrasi, menggunakan prinsip *self-service* dan *self-guided service*, menyusun dan memberikan isi pelajaran secara cepat, memberikan konsolidasi inisiatif pelatihan pada *platform web* yang terukur, menunjang portabilitas dan standarisasi, serta personalisasi isi dan memungkinkan penggunaan kembali pengetahuan yang sudah ada (Ellis, 2009, p. 2).

2.2.2. *Object Oriented Programming*

Object Oriented Programming (OOP) adalah sebuah pendekatan untuk pengembangan perangkat lunak dimana struktur perangkat lunak ini didasarkan pada benda-benda berinteraksi satu sama lain untuk menyelesaikan tugas. Interaksi ini membutuhkan bentuk pesan lewat bolak-balik antara objek. Dalam menanggapi pesan, objek dapat melakukan suatu tindakan atau metode. Contohnya seperti ketika seseorang ingin pergi ke toko dan berinteraksi dengan sebuah mobil. Sebuah objek mobil terdiri dari benda-benda yang berinteraksi satu sama lain untuk menyelesaikan tugas untuk mengantar orang tersebut ke toko. Orang tersebut meletakkan kunci dan menyalakannya. Hal tersebut mengirimkan pesan ke objek pemula, yang berinteraksi dengan objek mesin untuk menyalakan mobil. Sebagai supir, hal tersebut membuatnya terisolasi dari logika bagaimana suatu objek pada suatu sistem bekerja sama untuk menyalakan mobil. Orang tersebut hanya memulai urutan kejadian dengan menjalankan metode mulai dari objek pengapian dengan kunci, kemudian menunggu respon keberhasilan atau kegagalan (Clark, 2011, pp. 1-3).

Banyak pengembang *software* bisnis yang beralih ke metode OOP dan bahasa pemrograman untuk menyelesaikan berbagai masalah. Keuntungannya adalah sebagai berikut (Clark, 2011, p. 3):

a. Sebuah transisi yang lebih intuitif dari analisis model bisnis perangkat lunak ke model penerapan perangkat lunak.
b. Kemampuan untuk memelihara dan menerapkan perubahan dalam program yang lebih efisien dan cepat.
c. Kemampuan untuk lebih efektif menciptakan sistem perangkat lunak menggunakan proses tim, memungkinkan para ahli (*specialists*) untuk bekerja pada bagian dari sistem.
d. Kemampuan untuk menggunakan kembali komponen kode dalam program lain dan membeli komponen yang ditulis oleh pengembang pihak ketiga untuk meningkatkan fungsi dari program mereka dengan sedikit usaha.
e. Integrasi yang lebih baik dengan sistem komputasi terdistribusi yang digabungkan secara longgar.
f. Peningkatan integrasi dengan sistem operasi modern.
g. Kemampuan untuk membuat tampilan antarmuka yang lebih intuitif untuk pengguna.

2.2.3. *Framework*

Framework adalah sekumpulan perintah/fungsi dasar yang dapat membantu dalam menyelesaikan proses-proses yang lebih kompleks dalam bentuk

pola kerja tersebut disusun sedemikian rupa (secara umum) agar ke depan konsep ini bisa digunakan untuk menyelesaikan suatu tugas yang berbeda dalam sebuah koridor yang sama.

Sebelum kita menentukan *framework* yang akan kita gunakan, kita harus mengingat beberapa kemungkinan yang akan dihadapi (Pressman dan Lowe, 2009, pp24-25):

- *Requirements evolve over time*.
- *Changes will occur frequently*.
- *Time lines are short*.

2.2.4. ASP.NET

ASP.NET merupakan sebuah *framework* untuk membuat aplikasi *web* yang bersifat dinamis. Dikembangkan oleh Microsoft untuk memungkinkan para programer untuk membangun*web site*, *web application*, dan *web service* (Pressman dan Lowe, 2009, pp24-25).

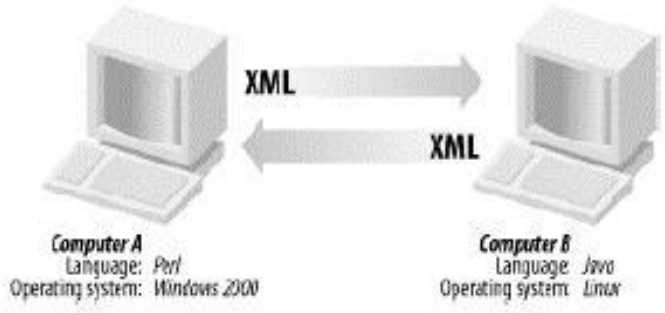
2.2.5. C#

C# merupakan bahasa berorientasi obyek yang sederhana, yang memungkinkan pemrogram untuk membangun aplikasi yang kompleks. Dikombinasikan dengan NET *Framework*., *Visual C#* 2008 memungkinkan pembuatan aplikasi *Windows*, *web service*, *database tools*, kontrol, komponen, dan yang lainnya. Sebagai bahasa berorientasi objek, C# mendukung konsep enkapsulasi, inheritance, dan polimorfisme. Semua variabel dan *method*, termasuk *main method*, *entry point* aplikasi, di-

encapsulate dalam *class definition*. Sebuah kelas dapat mewarisi secara langsung dari satu kelas induk (Microsoft, 2013).

2.2.6. *Web Service*

Web Service merupakan sebuah metode komunikasi antara dua perangkat pada *World Wide Web*. *Web service* menggunakan bahasa XML yang terstandarisasi dan tidak terikat oleh sistem operasi yang digunakan. (Cerami, 2002, pp. 6-7)



Gambar 2.6 *Web Service* Sederhana

Sebuah *web service* yang baik memiliki dua karakteristik yakni (Cerami, 2002, p. 7):

- Sebuah *web service* harus dapat mendeskripsikan isinya. Jika sebuah *web service* baru dibuat, maka perlu dibuat juga halaman yang berisi daftar *public method*, *method argument*, dan nilai kembalinya sehingga dapat mempermudah para pengembang untuk mengintegrasikan aplikasinya dengan *service* yang dibuat.

- Sebuah *web service* harus dapat dengan mudah ditelusuri. Jika sebuah *web service* dibuat maka pihak pengguna harus dapat menentukan lokasi *interface* dari *web service* tersebut dengan mudah.

Sifat-sifat dari *web service*:

- Tersedia pada jaringan *internet* atau *intranet*
- Menggunakan XML yang terstandarisasi
- Tidak terikat pada salah satu sistem operasi atau bahasa pemrograman
- Terdefinisi melalui XML
- Dapat ditelusuri melalui mekanisme yang mudah

2.2.7. *Visual Studio*

Visual Studio adalah *Integrated Development Environment* (IDE) di mana *developer* bekerja untuk membuat program dalam salah satu dari banyak bahasa, termasuk Visual C#. .NET *Framework* adalah lingkungan pengembangan dan pelaksanaan yang memungkinkan bahasa pemrograman yang berbeda dan librari untuk bekerja sama untuk membuat aplikasi *Windows*, *Web*, *Mobile*, dan *Office*. .NET *Framework* adalah suatu komponen *Windows* yang mendukung pembuatan serta menjalankan aplikasi generasi berikutnya dan XML *web service* (Microsoft, 2013).

2.2.8. *Hypertext Markup Language*

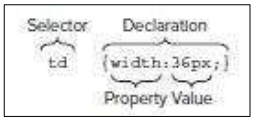
Hypertext Markup Languange (HTML) adalah bahasa *markup* utama untuk membuat halaman web dan informasi lain yang dapat ditampilkan di dalam sebuah *web browser*. HTML diperlukan dalam menentukan struktur dari sebuah halaman *web*, misalnya untuk menentukan bagian mana yang merupakan sebuah *header*, dimana sebuah paragraf dimulai dan diakhiri, di mana sebuah gambar akan ditampilkan, dan lainnya (Larsen, 2013, p. 1).

```
<html>
  <head>
    <title>Popular Websites: Google</title>
  </head>
  <body>
    <h1>About Google</h1>
    <p>Google is best known for its search engine, although
       Google now offers a number of other services.</p>
    <p>Google's mission is to organize the world's
       information and make it universally accessible and
       useful.</p>
    <p>Its founders Larry Page and Sergey Brin started
       Google at Stanford University.</p>
  </body>
</html>
```

Gambar 2.7 Contoh Struktur HTML

2.2.9. *Cascading Style Sheet*

Cascading Style Sheet (CSS) merupakan bahasa pemrograman untuk *web* yang digunakan untuk menentukan aturan-aturan bagaimana sebuah elemen ditampilkan di dalam halaman *web*. Aturan-aturan ini akan berpengaruh bagaimana isi dari elemen yang dimaksud akan diperlihatkan (Larsen, 2013, p. 192).



Gambar 2.8 Contoh Struktur Penulisan CSS

Pada gambar di atas menunjukkan sebuah contoh penulisan CSS yang terdiri menjadi dua bagian, yakni:

- *Selector* mengindikasikan elemen mana yang akan terpengaruhi oleh aturan yang akan diberikan. Elemen yang dimaksud dapat berjumlah satu atau lebih dari satu.
- *Declaration* menentukan bagaimana sebuah elemen yang ditentukan pada *selector* akan ditampilkan. *Declaration* terbagi menjadi dua bagian, yakni:
    - *Property* adalah jenis pengaturan pada elemen yang akan diatur, yang di mana pada contoh kali ini adalah *property width* yang akan mengatur bagaimana lebar dari sebuah elemen.
    - *Value* adalah spesifikasi dari *property*, yang pada kasus ini bernilai 36 satuan lebar.

2.2.10. *JavaScript*

JavaScript pada awalnya dikembangkan oleh Brendan Eich pada Netscape pada tahun 1995-1996. Saat itu JavaScript lebih dikenal dengan sebutan LiveScript. Javascript bukan merupakan bahasa pemrograman yang memerlukan kompilasi sehingga membuatnya menjadi bahasa pemrograman yang dapat diandalkan dan berguna untuk mensimulasikan dan menciptakan interaksi pada *World Wide Web* (Suehring, 2013, pp. 3-4).

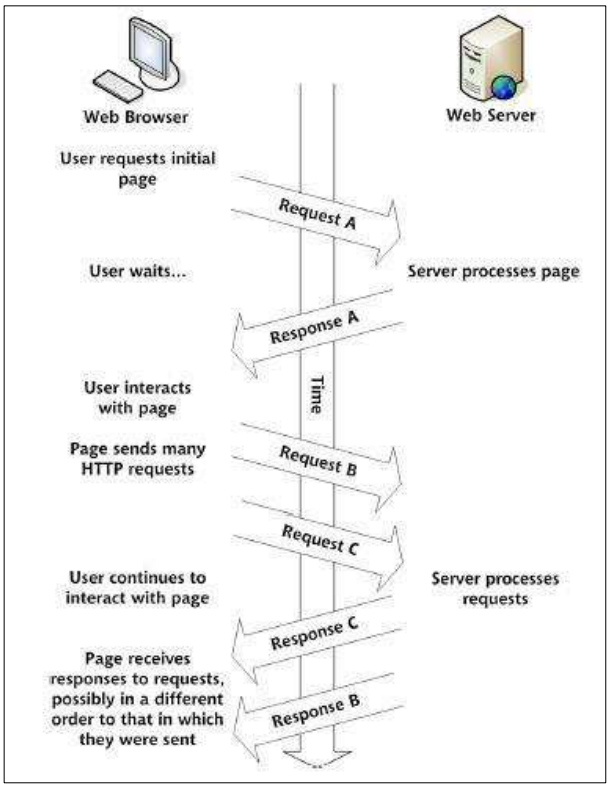
Hal-hal yang perlu diketahui mengenai JavaScript yakni (Valentine & Reid, 2013, p. 2):

- Merupakan bahasa *scripting*: JavaScript merupakan "scripts" yang dibaca dan dieksekusi oleh sebuah *interpreter* (atau *engine*). Hal ini yang membedakannya dengan bahasa kompilasi, di mana program dibaca oleh sebuah kompiler dan diterjemahkan ke dalam sebuah file *executable*. Program yang ditulis dalam bahasa *scripting* bersifat sangat*portable* karena dapat dijalankan pada berbagai *environment* dengan syarat terdapat *interpreter* padanya.
- Mirip dengan bahasa C: Sintaks dasar dan struktur JavaScript sangat mirip dengan C.
- Merupakan bahasa berorientasi objek: JavaScript berbeda dengan kebanyakan bahasa berorientasi objek lainnya, di mana model *inheritance* pada JavaScript menggunakan *prototype* dan tidak menggunakan *class.*
- Memiliki fungsi-fungsi utama: Fungsi-fungsi pada JavaScript merupakan objek utuh dan memiliki *properti* dan *method* tersendiri, dan dapat dikirimkan atau dikembalikan dari fungsi lain dan di-*assign* ke *variable*.
- Dinamis: Istilah "bahasa pemrograman dinamis" luas dan mencakup banyak fitur. Fitur JavaScript yang bersifat dinamis adalah pengimplementasian variabel dan fungsi *eval()* serta berbagai aspek fungsional lainnya.
- Variabel JavaScript tidak diperiksa pada saat waktu dilakukan interpretasi, dan bagaimana operasi terjadi antara *operand* yang berbeda tipe tergantung dari aturan-aturan spesifik yang terdapat pada JavaScript.

- Merupakan pengimplentasian sebuah standar yang ada: JavaScript merupakan implementasi dari standar ECMA-262, sama halnya dengan bahasa pemrograman C yang mengikuti standar ISO.

2.2.11. AJAX

AJAX, singkatan dari *Asynchronous JavaScript and XML*, merupakan sebuah teknik yang menggunakan JavaScript untuk melakukan *refresh* pada isi dari sebuah halaman dari sebuah *web server* tanpa harus melakukan *reload* keseluruhan halaman. Hal ini berbeda dengan cara *browser* dalam memperbarui isi halaman pada umumnya, yang mengharuskan *browser* melakukan *refresh* seluruh halaman untuk menampilkan perubahan pada isi halaman (Eernisse, 2006, p. 1).



Gambar 2.9 Sistem pada AJAX *web application*

2.2.12. JQuery

Jquery menyediakan sebuah*layer* abstraksi dengan berbagai fungsi dalam kaitannya untuk melakukan *scripting* web pada umumnya, dan oleh karena itu berguna dalam hampir seluruh situasi penulisan *script*. Sifatnya yang dapat ditambah sesuai kebutuhan sehingga penggunanya dapat menambahkan sendiri *plugin* yang yang memiliki fungsi lebih dari yang telah ada (Chaffer & Swedberg, 2013, pp. 8-9). Fitur utama dari jQuery adalah untuk membantu dalam pengerjaan hal-hal berikut:

- Mengakses elemen-elemen dalam sebuah dokumen: Tanpa sebuah *library* JavaScript, *web developers* seringkali diharuskan untuk menulis banyak

baris kode untuk menelusuri *Document Object Model* (DOM) *tree* dan menentukan lokasi bagian-bagian spesifik dari sebuah struktur dokument HTML. Dengan jQuery, *developer* memiliki mekanisma yang handal dan efisien dalam memperoleh elemen dokument yang ingin diinspeksi atau dimanipulasi.

- Memodifikasi tampilan dari sebuah halaman *web*: CSS menawarkan sebuah metode yang handal dalam menentukan bagaimana sebuah dokumen dirender, namun hal tersebut tidak berjalan semestinya ketika tidak semua *web browser* mendukung standar yang sama. Dengan jQuery, *developer* dapat menjembatani celah ini, dengan mengandalkan standar yang sama yang didukung oleh semua jenis *browser*.
- Mengubah isi dari sebuah dokumen: Tidak terbatas pada perubahan luar semata, jquery dapat memodifikasi isi dari dokumen itu sendiri dengan beberapa ketikan. Teks dapat diubah, gambar dapat dimasukkan atau diganti, *list* dapat disusun ulang, atau seluruh struktur dari HTML dapat ditulis ulang dan ditambah. Semua itu dapat dilakukan dengan memanfaatkan *Application Programming Interface* (API) yang mudah untuk digunakan.
- Merespon interaksi dari *user*: Fungsi atau fitur yang paling handal sekalipun tidak akan bermanfaat jika tidak dilakukan pada saat yang tepat. Jquery menawarkan cara untuk merespon berbagai variasi *event* yang ada, seperti ketika seorang *user* melakukan klik pada sebuah *link*, tanpa memerlukan perubahan pada baris HTML itu sendiri. Pada saat

yang sama, dukungan dari jQuery ini menghilangkan inkonsistensi *script* untuk setiap *browser* yang berbeda.

- Memberikan animasi ketika perubahan terjadi pada sebuah dokumen: Agar dapat secara efektif mengimplementasikan interaksi demikian, seorang desainer harus memberikan *visual feedback* kepada *user.* Jquery memfasilitasi hal ini dengan menyediakan berbagai efek seperti efek *fades* dan *wipes*, dan menyediakan perangkat untuk menciptakan efek lainnya.
- Memperoleh informasi dari sebuah server tanpa melakukan *refresh* pada halaman: Pola kode ini lebih dikenal dengan sebutan AJAX, singkatan dari *Asynchronous JavaScript and XML*, yang menawarkan teknologi handal dalam melakukan komunikasi antara *client* dan *server*. JQuery menghilangkan kompleksitas pada tiap *browser* dalam menggunakan fitur ini sehingga memungkinkan *developer* untuk lebih fokus pada fungsionalitas pada *server*.

  Mempermudah berbagai pekerjaan JavaScript pada umumnya: JQuery menyediakan tambahan kemampuan pada konstruksi JavaScript seperti fungsi iterasi dan manipulasi *array*.